AMENDMENTS TO THE CLAIMS:

Please amend claims 1, 11, 21, 22 and 26 as indicated in the following listing of claims. This listing of claims will replace all prior versions and listings of claims in the application:

1. (Currently Amended) A method of processing data comprising:

defining an object with defined fields to support values in preallocated memory space and with an option data structure which supports references to option values without preallocation of memory space for the full option values; ~~and~~

accessing a field value stored in one of the defined fields and accessing an option value not stored in the defined fields in the object using expressions of the same syntactic form; and

during compilation, determining whether at least one of the expressions accesses one of (a) a field value or (b) an option value;

when it is determined that the expression accesses a field value, compiling the expression into a first code for accessing the field value; and

when it is determined that the expression accesses an option value, compiling the expression into a second code for accessing the option value.

2. (Original) A method as claimed in claim 1 wherein the option data structure identifies change handler code that is executed when an option value changes.

3. (Original) A method as claimed in claim 2 wherein change handler code for one option is defined in different classes within a class inheritance hierarchy and the change handler code from each class is executed when the option value changes.

4. (Original) A method as claimed in claim 1 wherein the option data structure includes a type description of the option value, the method further comprising:

during compilation, using the type description in the option data structure to process an operation on the option value.

5. (Original) A method as claimed in claim 1 wherein an option data structure includes a default value, the method further comprising, in a get operation to an instance of the class, if an option value which applies to the instance has been set, getting the set option value and, if no value which applies has been set, getting the default value for the class.

6. (Original) A method as claimed in claim 1 comprising:

defining a first class with a first option data structure of a first form which supports, in instances of the class, references to option values without preallocation of memory space for the full option values;

defining a second class with a second option data structure of a second form which supports, in instances of the second class, references to option values without preallocation of memory space for the full option values, the second form being different from the first form; and

during compilation, encoding an option operation as a method call to an object of the first class and to an object of the second class without regard to the form of the option data structure supported by the class.

7. (Original) A method as claimed in claim 1 further comprising:

notifying objects of a change in an option value through a change handler identified by an option binding, the option binding being located by first searching a mapping data structure for a previously computed mapping to the option binding and, if no mapping was previously computed, by then computing the mapping to the option binding and storing the mapping in the mapping data structure.

8. (Original) A method as claimed in claim 1 wherein the option data structure comprises a linked list of option items having options values.

9. (Original) A method as claimed in claim 1 wherein a nonlocal option value applies to other objects in a nonlocal option hierarchy.

10. (Original) A method as claimed in claim 9 wherein the nonlocal option hierarchy is a graphical hierarchy.

11. (Currently Amended) A data processing system including data objects, the data objects comprising:

defined fields to support values in preallocated memory space;

an option data structure which supports references to option values without preallocation of memory space for the full option value, the field value stored in one of the defined fields and option value not stored in the defined fields being accessed in the object with expressions of the same syntactic form; and

a compiler which determines whether at least one of the expressions accesses one of (a) a field value or (b) an option value, wherein

when the compiler determines that the expression accesses a field value, the compiler compiles the expression into a first code for accessing the field value; and

when the compiler determines that the expression accesses an option value, the compiler compiles the expression into a second code for accessing the option value.

12. (Original) A system as claimed in claim 11 wherein the option data structure identifies change handler code that is executed when an option value changes.

13. (Original) A system as claimed in claim 12 wherein change handler code for one option is defined in different classes within a class inheritance hierarchy and the change handler code from each class is executed when the option value changes.

14. (Original) A system as claimed in claim 11 wherein the option data structure includes a type description of the option value, the system further comprising a compiler which uses the type description in the option data structure to process an operation on the option value.

15. (Original) A system as claimed in claim 11 wherein an option data structure includes a default value which is obtained when no option value has been set in an applicable instance object.

16. (Original) A system as claimed in claim 11 comprising plural classes having data structures of different forms, and a compiler which encodes an option operation as a method call to an instance object of one of the classes without regard to the form of the option data structure supported by the class.

17. (Original) A system as claimed in claim 11 further comprising change handlers which notify objects of a change in an option value and a mapping data structure which maps an option name and class to an option binding which identifies a change handler.

18. (Original) A system as claimed in claim 11 wherein the option data structure comprises a linked list of option items having option values.

19. (Original) A system as claimed in claim 11 wherein a nonlocal option value applies to other objects in a nonlocal option hierarchy.

20. (Original) A system as claimed in claim 19 wherein the nonlocal option hierarchy is a graphical hierarchy.

21. (Currently Amended) A data processing system comprising:

means for defining an object with defined fields to support values in preallocated memory space and with an option data structure which supports references to option values without preallocation of memory space for the full option values; ~~and~~

means for accessing a field value stored in one of the defined fields and accessing an option value not stored in the defined fields in the object using expressions of the same syntactic form; and

means for determining, during compilation, whether at least one of the expressions accesses one of (a) a field value or (b) an option value;

when it is determined that the expression accesses a field value, compiling the expression into a first code for accessing the field value; and

when it is determined that the expression accesses an option value, compiling the expression into a second code for accessing the option value.

22. (Currently Amended) A computer program product comprising:

a computer usable medium for storing data; and

a set of computer program instructions embodied on the computer usable medium, including instructions to:

define an object with defined fields to support values in preallocated memory space and with an option data structure which supports references to option values without preallocation of memory space for the full option values; ~~and~~

access a field value stored in one of the defined fields and access an option value not stored in the defined fields in the object using expressions of the same syntactic form; and

during compilation, determine whether at least one of the expressions accesses one of (a) a field value or (b) an option value;

when it is determined that the expression accesses a field value, compiling the expression into a first code for accessing the field value; and

when it is determined that the expression accesses an option value, compiling the expression into a second code for accessing the option value.

23. (Original) A product as claimed in claim 22 wherein the computer program instructions include instructions to notify objects of a change in an option value.

24. (Original) A product as claim in claim 22 wherein the option data structure comprises a linked list of option items having option values.

25. (Cancelled).

26. (Currently Amended) A method of processing data comprising:

defining one or more objects from a class definition, each of the objects having defined fields to support values in preallocated memory space and having an option data structure which supports references to option values without preallocation of memory space for the full option values;

accessing a field value stored in one of the defined fields and accessing an option value not stored in the defined fields in the object using expressions of the same syntactic form; ~~and~~

identifying change handler code that is executed when an option value stored in the option data structure changes, wherein the change handler code is associated with the class definition during compilation; and

during compilation, determining whether at least one of the expressions accesses one of (a) a field value or (b) an option value;

when it is determined that the expression accesses a field value, compiling the expression into a first code for accessing the field value; and

when it is determined that the expression accesses an option value, compiling the expression into a second code for accessing the option value.

27. (Previously Presented) The method of claim 26, wherein change handler code for one option is defined in different classes within a class inheritance hierarchy and the change handler code from each class is executed when the option value changes.

28. (Previously Presented) The method of claim 26, wherein the option data structure includes a type description of the option value, the method further comprising:

during compilation, using the type description in the option data structure to process an operation on the option value.

29. (Previously Presented) The method of claim 26, wherein an option data structure includes a default value, the method further comprising:

in a get operation to an instance of the class, if an option value which applies to the instance has been set, getting the set option value and, if no value which applies has been set, getting the default value for the class.

30. (Previously Presented) The method of claim 26, further comprising:

defining a first class with a first option data structure of a first form which supports, in instances of the class, references to option values without preallocation of memory space for the full option values;

defining a second class with a second option data structure of a second form which supports, in instances of the second class, references to option values without preallocation of memory space for the full option values, the second form being different from the first form; and

during compilation, encoding an option operation as a method call to an object of the first class and to an object of the second class without regard to the form of the option data structure supported by the class.